JOINT INVENTORS

"EXPRESS MAIL" mailing label No. EM578403920US.
Date of Deposit: September 1, 1999
I hereby certify that this paper (or fee) is being deposited with the United States Postal Service "EXPRESS MAIL POST OFFICE TO ADDRESSEE" service under 37 CFR §1.10 on the date indicated above and is addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231

[illegible]

G. Christopher Braidwood

APPLICATION FOR UNITED STATES LETTERS PATENT

S P E C I F I C A T I O N

TO ALL WHOM IT MAY CONCERN:

Be it known that we, Mark G. Dreyer, a citizen of the United States of America, residing at 2320 Blue Spruce Lane, Aurora 60504, in the County of DuPage and State of Illinois; Riyaz M. Asaria, a citizen of the United States of America, residing at 202 Pleasant Street, Oak Park 60302, in the County of Cook and State of Illinios; J. Thomas Shively, a citizen of the United States of America, residing at 45 S. Thurlow Street, Hinsdale 60521 in the County of DuPage and State of Illinois; and James L. Warmus, a citizen of the United States of America, residing at 350 S. Kensington, LaGrange 60525 in the County of Cook and State of Illinois, have invented a new and useful SYSTEM AND METHOD FOR AUTOMATED CLOSED-LOOP PRODUCTION OF CUSTOMIZED BOOKS, of which the following is a specification.

SYSTEM AND METHOD FOR AUTOMATED CLOSED-LOOP PRODUCTION OF CUSTOMIZED BOOKS

FIELD OF THE INVENTION

The present invention relates generally to book production methods and systems, and more particularly to a method of and system for automated closed-loop production of customized books.

BACKGROUND OF THE INVENTION

Printed documents are often assembled into books, such as catalogs, brochures, manuals, and the like. Such documents may have diverse content including text, images, and line art positioned in a myriad of ways. Through the use of page make-up software, each page of a book may be designed to include such content using a computer. In the specific example of printed catalogs, the page make-up software is used to design the pages of the catalog such that the pages include, for example, product names, descriptions, images, and prices.

The design of a catalog (or any other type of book) often begins with the creation or development of content, which includes both textual data and image portions. Traditional preliminary prepress steps, such as color correction, undercolor removal, and screening, may then be performed to prepare the content for incorporation into a page description file, such as a QuarkXPress® file. For a catalog, the page description file may contain detailed information for each of a number of products spread over one or more pages. In addition to this content-related information, the page description files usually contain further information directed to the layout of each page, *i.e.*, formatting and other control data. Taken together, the content information and control information are supplied to a raster image processor or other apparatus responsive to page description files for processing in connection with either a computer-to-plate or digital press workflow.

Generation of these page description files has typically been a labor-intensive process. A publisher or other layout designer can spend a great deal of time manually gathering, preparing, and formatting the content for a single book despite the efficiencies provided by the page make-up software. Even if the majority of the content of a book has been incorporated into a previously published book, extensive time may still be devoted to page layout and other formatting issues.

The time available for layout design and related processing is particularly limited for certain types of catalogs. Seasonal clothing catalogs, for example, are typically printed and distributed several times each year, often with clearance catalogs interspersed between the seasonal versions. However, the marketing of clearance items through special editions of a catalog usually depends upon timely inventory determinations. In these situations, the layout designer may proceed through the laborious process of developing the page description file(s) for the clearance catalog, only to find out that the inventory has changed dramatically in the interim (*e.g.*, some products have sold out). Clearance catalogs designed and processed in current production workflows are, thus, routinely outdated because of the finite amount of time needed to generate the page description files and the ongoing sales during such time. Such outdated catalogs have often been forced to rely on red-lining or other undesirable modifications to accurately indicate which products or items are no longer available.

SUMMARY OF THE INVENTION

In accordance with one aspect of the present invention, a software system is useful for generating, from a first page description file having portions wherein each portion includes content, a second page description file. The software system includes a computer-readable medium and a routine stored therein. The routine includes a first routine that provides for generation of a template, a second routine that provides for extraction of data indicative of the portions of the first page description file to generate a database for storing the data, and a third routine that generates the second page description file from the template and the database.

According to a preferred embodiment, the first routine provides a user interface for establishing placeholders in the template. The user interface preferably includes a page make-up software application and a placeholder definition routine for establishing the placeholders in the template. The page make-up software application may be QuarkXPress®.

According to another preferred embodiment, the first routine generates the template in accordance with the portions of the first page description file. In yet another preferred embodiment, the second routine provides a user interface for characterizing each portion of the portions of the first page description file as an instance of at least one field of a plurality of fields of the database.

Each portion of the first page description file may include a first data portion and a second data portion. The second routine then preferably provides a user interface for parsing each portion of the first page description file to separate the first data portion from the second data portion. Further, the first data portion of each portion may include content data and the second data portion of each portion may include control data.

Preferably, the user interface of the second routine stores the first data portion of each portion in the database.

In still another embodiment of the present invention, the template is associated with a plurality of pages to be printed in a book. The template may be one of a plurality of templates such that each record of the database comprises information indicative of a certain template of the plurality of templates.

Preferably, the routine includes a fourth routine that provides for modifying the data stored in the database prior to generation of the second page description file. In this embodiment, the fourth routine provides a user interface for modifying the data stored in the database.

In accordance with another aspect of the present invention, a method is useful for generating, from a first page description file having portions wherein each portion includes content, a second page description file. The inventive method includes the steps of generating a template, extracting data indicative of the portions of the first page description file to generate a database that stores the data indicative of the portions of the first page description file, and generating the second page description file from the template and the database.

In a preferred embodiment, the template generating step includes the step of establishing placeholders in a file developed in a page make-up software application. Alternatively, the template generating step generates the template in accordance with the portions of the first page description file.

In another preferred embodiment, the data extracting step includes the step of characterizing each portion of the first page description file as an instance of at least one field of a plurality of fields of the database.

Each portion of the first page description file may include a first data portion and a second data portion. The data extracting step may then include the step of parsing each portion of the first page description file to separate the first data portion from the second data portion. The first data portion of each portion may include content data and the second data portion of each portion comprises control data. The data extracting step then preferably includes the step of storing the first data portion of each portion in the database.

According to still another preferred embodiment, the inventive method further includes the step of modifying the data stored in the database to reflect modified content such that the second page description file generating step comprises the step of generating the second page description file from the modified data stored in the database.

Other features and advantages are inherent in the system and method claimed and disclosed or will become apparent to those skilled in the art from the following detailed description in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a sample page of a catalog as represented by a page make-up program, such as QuarkXPress®;

FIG. 2 illustrates a main dialog box which is displayed to an operator during the process of converting page make-up files to page description files;

FIG. 3 is a high-level flowchart of a portion of the process of converting page make-up files to page description files;

FIG. 4 is a flowchart of programming executed by the block 30 of Fig. 3;

FIGS. 5A-5C, when joined side-by-side with Fig. 5A on the left, Fig. 5B in the middle and FIG. 5C on the right, is an illustration of the product table noted in FIG. 4;

FIG. 6 is an illustration of the SKU table noted in FIG.4;

FIG. 7 is a flowchart of programming executed by the block 32 of FIG. 3;

FIG. 8 is a flowchart of programming executed by the block 34 of FIG. 3;

FIG. 9 is a flowchart of programming executed by the block 36 of FIG. 3;

FIG. 10 is an illustration of the dialog box of FIG. 2 with sample information entered therein;

FIG. 11 is a flowchart of programming executed by the block 154 of FIG. 9;

FIGS. 12 and 13 are illustrations of dialog boxes which are presented to an operator during execution of the programming of FIG. 11;

FIG. 14 is a flowchart of programming executed by the block 198 of FIG. 11;

FIG. 15 is a flowchart of programming executed by the block 158 of FIG. 9;

FIG. 16 is an illustration of a dialog box which is presented to an operator during execution of the programming of FIG. 15;

FIG. 17 is a flowchart of programming executed by the block 162 of FIG. 9;

FIG. 18 is a flowchart illustrating the a portion of the programming executed by the block 166 of FIG. 9 when the "Save" button of FIG. 10 is selected;

FIGS. 19A-19C, when joined side-by-side with FIG. 19A on the left, FIG. 19B in the middle and FIG. 19C on the right, is an illustration of the product table with sample data stored therein;

FIGS. 20A-20C, when joined along similarly lettered lines, together represent programming executed by the block 38 of FIG. 3;

FIG. 21 is an illustration of a dialog box which is presented to an operator during execution of the programming of FIGS. 20A-20C;

FIG. 22 is a flowchart of programming executed by the blocks 432 and 424 of FIGS. 20A and 20C, respectively;

FIG. 23 is a block diagram of a computer system which may be used to implement the present invention;

FIG. 24 is a block diagram of a process of converting print-ready documents to web pages according to the present invention; and

FIG. 25 is a workflow diagram that illustrates the incorporation of portions of the process of FIG. 3 into a closed-loop production system and method in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

The present invention is described below with reference first to the conversion of the contents of QuarkXPress® print-ready documents into one or more Microsoft Excel® database tables and image(s) converted into a web-ready (*i.e.*, Internet) format using Adobe Photoshop®. The contents of the database tables can then be converted to a web format and the resulting text coding and web-ready image(s) can then be formatted by a display program known as Marketplace (Version 2) licensed by Multimedia Live of Petaluma, California, to dynamically generate web pages for display by a server (not shown) on the Internet (FIG. 24 illustrates this process). However, it should be apparent to one of ordinary skill in the art that the conversion process and utility described herein may be used in other applications or with other page layout programs, image manipulation programs and/or page development programs and still be within the scope of the present invention.

In accordance with the present invention, the database tables and images utilized to develop the web pages may be utilized as part of a closed-loop production workflow in which the database tables and images are utilized to develop page description files for production of customized pages in any target or output medium, such as a computer monitor, ink or toner on paper, dye in film, and the like. Also, while the present invention is described in the context of converting and processing the pages of a print-ready catalog, it should be understood that other printed matter may instead be processed by the present invention.

According to the workflow described herein, once copies of the catalog are distributed (whether via hardcopy, electronically or otherwise), orders for items presented in the catalog are processed that modify the inventory of products presented in the catalog. The system and method of the present invention may utilize data indicative of this modified inventory and any other information relating to the processed orders to facilitate the generation of further catalogs having customized content. In this manner, the system and method of the present invention provide a user with a closed-loop workflow for producing printed (or otherwise distributed) matter having content that changes as a result of sales originating from the distribution of previous catalogs.

Initially, one or more page make-up files (also referred to as layout files) are created using a computer system running a page make-up (or layout) program, such as QuarkXpress®. The computer system may be like that shown in FIG. 23 including a computer 25 having internal memory 26, input devices 27, such as a keyboard and mouse, a monitor 28 and, optionally, a connection to a network 29. The page make-up file(s) represent a book which has or will be produced, such as a catalog or other printed matter. Generally, these page layout file(s) can be stored on a computer-readable memory such as a hard drive or other storage device or the page layout file(s) may be stored and accessed remotely through the network 29. FIG. 1 illustrates a sample page from a seed catalog as displayed by QuarkXpress®. The sample page includes three catalog items (alternatively termed objects herein) referred to by the large text "Sugar Crunch," "Watercolor Memories" and "Super Tasty." All catalog items contain a text portion and a high resolution image. The text portion for each catalog item contains detailed information including item name, description, catalog number or SKU number, style description and price. Other information and variations on this detailed information may be provided. Furthermore, not every catalog item need have the same types or levels of detailed information.

The text portions of the page layout file are untagged and unstructured, *i.e.*, they are not stored as separate character strings based on the contents thereof but rather as a single group of characters. Thus, while the text "Sugar Crunch" is the name of one of the catalog items, "B-54031" is the item number and "$2.95" is the price for 30 seeds, all of this text is stored as one complete group of characters in the page layout file, with embedded characters indicating special symbols, fractions, end-of-line, etc... Furthermore, these text items are stored in the page layout file without relational linking among the text items, e.g., the price string for "Sugar Crunch" is not relationally linked to the item number string even though they both relate to the same catalog item or object.

An automated conversion process converts the text portions of the page layout file into a database and converts the image(s) into a proper format for generation of web pages or another type of visual display. A main dialog box for the conversion program is shown in FIG. 2 and is opened by a user operating a computer system like that shown in FIG. 23. The page layout file is also opened using the native application that created the page layout file (in the illustrated example, QuarkXpress®). The dialog box includes fields for substantially all of the text items relating to a given catalog item. Each dialog box field contains a label indicating the field name (i.e., "Item Number," "Item Name," "Headline," "Tn Description," (for thumbnail description) "Image Name," "Keywords," "Description," "Detail Image(s)," "Category Information" and "SKU Information") and a text box into which converted text from the page layout file or information generated by the user is placed and stored. The dialog box fields preferably coincide with the contents of the catalog items in the page layout file, and therefore may vary from catalog to catalog. In the present example, the fields identified by the labels "Item number," "Item name," "Headline," and "Description" are referred to as conversion fields because text from the page layout file is automatically converted to a desired format and cleaned (i.e., undesired codes are removed and other codes are inserted to cause proper display of text) by a conversion program when the text is placed into such fields. This conversion and cleaning can be performed in a manner unique to each conversion field or some or all of the conversion fields can employ the same conversion and cleaning process. The remaining fields are referred to as user fields because the information stored therein is provided by the user (except for some fields in the "SKU Information" area), as described in detail below.

With reference to the generalized flow diagram of FIG. 3, the steps shown therein may be executed by a user using the computer system of FIG. 23 after the dialog box of FIG. 3 and the page layout file are opened. The steps of FIG. 3 and the flowcharts following FIG. 3 may be partially or fully implemented by software programmed in a desired programming language, such as AppleScript® by Apple Computer, Inc. At least a portion of the programming described herein is set forth in the AppleScript® coding included in the Appendix attached hereto. After initialization of process variables and data structures, a block 30 opens a database file using a native application, such as Excel®. The database file comprises a template which stores converted and cleaned data from the conversion fields of the dialog box of FIG. 2. Next, a name translation file is opened by a block 32. The name translation file stores the names of images in the page layout file that are to be processed. Thereafter, a categories file, which contains previously stored information listing possible categorizations of a particular catalog item, is opened and processed by a block 34, again using a native application, such as Excel®. In summary, the blocks 30, 32 and 34 open files that will be accessible during use of the dialog box shown in FIG. 2. Next, at a block 36, the user can select text displayed by the page make-up program by highlighting such text using the mouse and clicking on one of the conversion fields of the dialog box of FIG. 2. This action causes the block 36 to run the conversion program to convert and clean the text selected from the page layout file. The converted and cleaned text is also displayed in the selected conversion field.

After the block 36, a block 38 runs an image conversion program which converts any image(s) into a format suitable for display.

Though the process steps of FIG. 3 are shown in a particular order, one of ordinary skill in the art will appreciate that the first three steps 30, 32 and 34 may be performed in any order and that the steps 36 and 38 may be undertaken independently. Furthermore, it should be appreciated that any of the steps may be initiated from the dialog box of FIG. 2. By way of example, the categories file could be opened from a specified directory by clicking on one of the boxes of the category information field of the dialog box of FIG. 2.

A detailed flow diagram of the programming executed by the block 30 of FIG. 3 is shown in FIG. 4. A block 62 displays a dialog box prompting the user to open a new or existing Excel® database file. Control pauses at a block 64 until a database file is selected. Each database file is preferably organized into multiple tables to separate different types of information. For example, a database file may contain a product table 66, a SKU table 68, an alternate image table 70, etc... The database file may also include scratch tables which temporarily store data. Each table comprises a plurality of records, and each record includes a number of fields each organized under a field heading. A sample empty product table is shown in FIGS. 5A, 5B and 5C. The product table has item number, item name, keywords, category, manufacturer name, manufacturer logo, main category 1 and 2, sub category 1 and 2, detail category 1 and 2, headline, thumbnail description, thumbnail image, product description, product page image, publish, default price, and tax exempt headings. These headings are exemplary in nature and may be replaced by other headings depending upon the particular catalog or other printed matter that is being converted for display. Other field headings may also be used depending upon whether the page layout file contains more information to be characterized and displayed. Preferably, there is at least one product table heading for each of the conversion fields of the dialog box of FIG. 2.

A detailed view of the SKU table is shown in FIG. 6. The SKU table is used to track items for availability and promotion. The SKU table may have, for example, a heading for promotion name, promotion description, promotion rank, start and end dates of promotion, etc...

The alternative image table is used to track "detail images." A detail image is an image derived from the high resolution image of an object. For example, a detail image may be a cropped and enlarged portion of a high resolution image.

Referring again to FIG. 4, once the user has selected a database file, a block 66 opens the tables of the file and a block 68 searches for the next available open (or blank) row in each table. Because these rows may not be in the same location in the various tables, the locations of these next available rows are noted and stored by the block 68 so that all of the information for a given object that is populated into the database records will be linked together. After the next open row for each table is found, the rows are marked for data entry by the block 68 and thereafter the database file selection process ends.

FIG. 7 illustrates the programming executed by the block 32 of FIG. 3. The name translation file is a file which maps the names of the high resolution image files in the page layout file to the names of images that will be created during the image processing of the block 38 of FIG. 3. As discussed below, the high-resolution images of the page layout file, which vary in size and shape, may be converted in an automated fashion into images that can be displayed in or on the target medium (e.g., the Internet) including a thumbnail image and/or a cropped image. The name translation file stores the names of all images that are noted in the fields labeled "Detail Image(s)" of the dialog box of FIG. 2.

A block 90 displays a dialog box (not shown) prompting the user to select opening of either an existing or a new name translation file. If the user selects that an existing file is to be opened, which would be done to continue inputting image names from a previously saved catalog conversion, a block 92 displays a dialog box (also not shown) which presents all existing file names for selection by the user. Once the user indicates selection of a particular file, a block 94 opens the selected file and determines the next empty row in the table. Control then passes to the block 34 of FIG. 3. If the operator opens a new file, a block 96 displays a dialog box (not shown) which prompts the user to enter a file name and a block 98 opens the file. A block 100 then sets the file length to a null value so that all data that may be in the file are cleared. Control then passes to the block 34 of FIG. 3.

The block 34 of FIG. 3 processes a category file created by the same user or a different user. Generally, an object can be categorized to permit the objects to be searched at a later time. Each object can be categorized at three levels under the "Main," "Sub," and "Detail" headings of the fields labeled "Category Information" of the dialog box of FIG. 2. In order to facilitate this categorization, the category file is processed to provide the possible listings for a main category, a sub-category and a detail category. A sample format for a category file is listed in TABLE 1 below:

TABLE 1

C	S	D	STR
1	0	0	Lawn & Garden
1	1	0	Flowers
1	1	1	Geraniums
1	1	2	Marigolds
1	1	3	Sunflowers

1	1	4	Pansy
1	1	5	Phlox
1	1	6	Iris
1	1	7	Daisy
1	2	0	Vegetables
1	2	1	Tomatoes
1	2	2	Squash
1	2	3	Gourds
1	2	4	Melons
1	2	5	Beans
1	2	6	Cucumbers
1	2	7	Swiss Chard
1	2	8	Onions
1	2	9	Eggplant
2	0	0	Toys & Games

TABLE 1 includes a variable C identifying a number of possible main categories, a variable S identifying a number of possible sub-categories and a variable D identifying a number of possible detail categories. The fourth column of TABLE 1 contains a string STR identifying the name or title of the corresponding categorization. For example, the first entry in the table includes the string "Lawn & Garden." Because this entry has a zero value for the sub-category variable S and the detail variable D, this string is a main category title. Similarly, "Toys & Games" is a main category title, as indicated by the stored values of C=2, S=0, and D=0. There are two sub-categories under the "Lawn & Garden" main category identified as "Flowers" and "Vegetables," as indicated by the zero value for the variable D. Under the "Flowers" sub- category, there are numerous detail categories, such as "Geraniums," "Marigolds," "Sunflowers," "Pansies" and "Phlox." Detail categories such as "Tomatoes," "Squash," "Beans" and "Cucumbers" are arranged under the "Vegetables" sub-category (which, in this case, results in an incorrect categorization because tomatoes are a fruit and beans are legumes). The category file contains as many main categories, sub-categories and detail categories as are desired by the user or called for by the design of the web pages. Preferably, there are at least enough categories to adequately describe all of the objects in the print-ready file.

FIG. 8 illustrates the programming executed by the block 34 of FIG. 3. A block 120 displays a dialog box which prompts the user to select a categories file for processing. Once a file is selected by the user, the categories file is opened by a block 122 and a block 124 checks to determine whether all records in the categories file have been processed. If not, a block 126 assigns the values of the next record in the file to variables C, S, D, and STR. A block 128 then checks to determine whether the value of S is equal to zero. If this is the case, a block 130 stores the string STR in memory as a main category. If the block 128 determines that the value of S is not equal to zero, a block 132 determines whether the value of the variable D is equal to zero. If this is true, a block 134 stores the string STR in memory as a sub-category under the main category identified by the values C=X, S=0 and D=0, where X is the value of C for the current record.

If the block 132 determines that the value of D is nonzero, a block 136 stores the string STR as a detail category in memory under the subcategory identified by the values C=X, S=Y and D=0, where Y is the value of S for the current record.

It should be evident from the foregoing that the programming of FIG. 8 is dependent upon the format of the categories file and that such programming may change if the categories file format is different from that set forth above.

With the pre-conversion steps complete, the block 36 of FIG. 3 performs the conversion and cleaning process. A detailed description of the programming executed by the block 36 is presented in FIG. 9.

Initially, a block 150 displays the main dialog box of FIG. 2. The user then can highlight a portion of the text on the page layout file and click on a corresponding one of the conversion fields. Alternatively, the user can click on any other field or button without first highlighting text in the page layout file. Once an action is taken by the user, a series of blocks 152, 156, 160 and 164 determine whether the user has clicked on a field in the "SKU Information," "Category Information" or "Detail Image(s)" areas or on the "Image Name" field or on one of the buttons at the bottom of the dialog box. If any of these conditions is found to be true, control passes to one of blocks 154, 158, 162 or 166. The programming executed by blocks 154, 158 and 162 is shown in detail in FIGS. 11, 15 and 17, respectively. In the case of the block 166, clicking on the button labeled "Reset Excel" causes the database file opened by the block 30 of FIG. 3 to be reset, i.e., the contents of the database file are deleted from the table. Clicking on the button "Save Excel" causes the information in the database file to be saved. Clicking on the button "Edit on SKU WS" causes the Excel® program to be displayed with the SKU table opened to permit direct editing thereof. Clicking on the button "Save SKU WS" causes the SKU table to be saved. If the user clicks on the "Reset" button, the contents of all the fields of the dialog box of FIG. 3 are erased. If the user clicks on the "Save" button, the data in the various fields of the dialog box are saved to the appropriate files. Clicking on the "Done" button causes the information in the fields of the dialog box to be saved to the appropriate files and closure of the dialog box.

If none of the conditions checked by the blocks 152, 156, 160 or 164 is found to be true, then it has been determined that the user has clicked on one of the conversion fields, and hence a block 168 automatically converts and cleans the text that was highlighted by the user. The highlighted information is filtered to remove and/or replace characters which, if unaltered, would not display properly in some or all web browsers. Control from each of the blocks 154, 158, 162, 166 and 168 returns to the block 150.

Preferably, all of the information for a specific conversion object is characterized (by highlighting specific text and clicking on the appropriate conversion field) and cleaned before converting information for the next conversion object. For example, referring again to FIG. 1, all of the information for the object "Sugar Crunch" is characterized and cleaned before processing is undertaken for the object "Watercolor Memories."

FIG. 10 illustrates the dialog box of FIG. 2 with information entered for the "Sugar Crunch" object. The text strings in the "Item Number," "Item Name," "Headline" and "Description" fields are separately entered into such fields, at which time the strings are converted and cleaned. The text strings in the "Tn Description," "Keywords" and "Detail Images" fields are entered by the user. The information in the "Category Information" fields is entered by successively clicking on the fields under the "Main," "Sub" and "Detail" headings, in turn causing selections for such fields to be displayed by a pop-down menu. The appropriate information may then be selected by the user by clicking on same.

Some of the information in the "SKU Information" field is entered like the information in the conversion fields (at which point such data are cleaned and converted) while the remaining information is manually entered by the user. Before describing this process further it is helpful to describe the ways in which an object is identified. In the illustrated example, each catalog object is identified by at least one item number which is stored in the "Item Number" conversion field. In addition to the item number, a catalog object is identified by at least one SKU number. Usually, an object is identified by more than one SKU number when the object may have differing attributes. For example, different SKU numbers may denote seeds which are for the same plant or vegetable except that the seeds have variations in style, colors, or sizes. Thus, seeds which grow into a blue version of a flower may have a different SKU number than seeds which grow into a red version of the same flower. The "SKU Information" area of the dialog box includes fields which allow object attributes and other information relating to an object to be specified. The fields include SKU, Style, Price, Color, Size, Sale Price ("Sale P"), Quantity Price("Quan"), Quantity Needed ("Quan N"), Thumbnail Price ("TP"), Weight ("W") and M. (When a "Y" is stored in the M field, the data in the respective row of the "SKU Information" table are stored in the SKU table as marked text, which may be in a different color than the remaining data in the SKU table. This facilitates location of such data so that the data can be further processed as desired.) Other fields may be used depending on user preference and/or catalog content.

The programming executed by the block 154 of FIG. 9 is shown in detail in FIG. 11. The programming is initiated by clicking on one of the "SKU Information" fields of the dialog box of FIG. 2. A block 190 then opens a dialog box as illustrated in FIG. 12. This dialog box contains fields for SKU Number, Style, Price, Color, Size, Sale Price, Quantity Price, Quantity Needed, and Weight, all of which have corresponding fields in the main dialog box of FIG. 2. Once the user clicks on a field or button of the dialog box of FIG. 12, a block 192 determines whether a button denoted "Combos" has been clicked on. If this is true, a block 194 displays the dialog box illustrated in FIG. 13 to allow the user to specify combinations of attributes (this operation is discussed in greater detail hereinafter.) Control then returns to the block 190. If the block 192 determines that the "Combos" button has not been clicked on, a block 196 determines whether one of the text fields has been clicked on. If this is true, a block 198 executes the routine executed in FIG. 14. This allows the user to enter SKU data through the use of the highlight text process described above with respect to the inputting of data into the conversion fields. Thus, if a page layout file has SKU information the user may highlight that information within the page layout file and click on the respective SKU conversion field to convert and clean the data in the same manner as described above and then store clean data into the SKU conversion fields.

If the block 196 determines that one of the text fields has not been clicked on, a block 200 determines whether an "OK" button has been clicked on. If this is the case, the data in the text boxes are transferred to the fields of the "SKU Information" area of the dialog box of FIG. 2 by a block 201. Control then passes to the block 150 of FIG. 9. Otherwise, a block 202 executes programming which checks to determine whether a "Thumbnail Price" box, an "Apply" button, a "Copies" field, a "Delete" button, a "Mark Item" box or a "Cancel" button has been clicked on. If the "Thumbnail Price" box has been clicked on, then a value is loaded into a field referred to as "Thumbnail Price" in the database. If the "Apply" button has been clicked on, then the values in the "Item Number," "SKU Number," "Size," "Sale Price," "Quantity Price," "Quantity Needed," Weight" and "Thumbnail Price" fields are loaded into temporary memory for use during "Combo" processing, as noted hereinafter. If the "Copies" field has been clicked on, the user may specify that the information in text fields may be copied a specified number of times into the SKU table. If the "Delete" button has been clicked on, the corresponding record in the SKU table is deleted. When the "Mark Item" box has been clicked on, the entry in the SKU table corresponding to the information in the text boxes is marked as noted above. If the "Cancel" button has been clicked on, the data in text fields (if any) are deleted. Control from the block 202 then returns to the block 190.

Alternatively, if desired, SKU data may be automatically stored into the SKU text fields through the use of a combinations (hereinafter "combo") selection process initiated by clicking on the "Combos" button of the dialog box of FIG. 12. Once this button is clicked on, the dialog box of FIG. 13 is displayed. At this point the user can copy multiple entries from combination tables by clicking a "Load" button under each of three columns of text boxes. These combination tables may be previously generated by the same or a different user. Each combination table includes a plurality of columns each including a different combination of attribute values. The text boxes may be filled as shown in the example of FIG. 13. If the user decides to clear the entries in the text boxes, he or she can click on one or more of the "Clear Style," "Clear Color" or "Clear Size" buttons or the user can directly create and/or edit the entries. The user can click on the "Make Combos" button to generate a combination table listing all possible combinations of the various attribute values in the text boxes along with the parameters stored in temporary memory when the "Apply" button was selected. The "Save Combos" button can then be selected to save the combination table. If desired, the "Advanced..." button can be clicked on to open the combination table in Excel® so that more powerful editing features can be employed. When the user is finished with the combo process, he or she can select the "Done" button, whereupon the dialog box of FIG. 13 is closed.

It should be noted that the design of the dialog box of FIG. 13 will be different if values of different attributes are to be combined.

FIG. 14 illustrates the converting and cleaning process invoked by the block 168 of FIG. 9 and the block 198 of FIG. 11. A block 240 fetches the highlighted text from the page layout file. Next, a block 242 converts and cleans the text. In the illustrated embodiment, text cleaning software routines are written as a number of AppleScript® routines, each performing a different cleaning function. First, the block 242 can include a character counting function to truncate strings longer than a predetermined length. For example, if it is desirable to limit the product description records of the product table to only 30 characters to ensure proper display in a web page, then the block 242 could include a character counter which truncates any characters of the highlighted text of the page layout file that are beyond the thirtieth character. This character counting and truncation routine can be programmed using known routines. Alternatively, the user could be prompted to reselect text from the page layout file.

Second, the block 242 may include an embedded control character elimination routine. In order to accomplish this cleaning function, an AppleScript® routine is written which stores the characterized text into a character string. The embedded character elimination routine iterates through each character within the string. As it iterates through each character, the routine determines if the ASCII value for the current string character is between a user-definable range of ASCII values. This range of ASCII values corresponds to the range of ASCII characters which will be properly displayed on-line. Characters outside of this desirable range, such as control characters, are removed or replaced with a more appropriate character (e.g., a tab may be replaced by a single space) by this second cleaning routine. As characters from the stored highlighted string are removed, the string is truncated by the removal. Alternatively, the embedded character elimination routine could compare the ASCII value of each character to a user-specified set of ASCII characters, instead of to a user specified range of ASCII characters, and remove those ASCII characters which match the user-specified ASCII characters.

Certain ASCII characters, like ®, ©, and ™ should not be removed, but rather replaced with codes recognizable by a web browser to display the character properly on-line. For example, the conversion program may contain routines which identify these ASCII characters and replace them with command strings recognizable in HTML, like &reg;, &copy; and &trade;, respectively. Other characters may be replaced through similar routines. In addition to these cleaning routines, any number of other cleaning routines, such as removing double spaces from highlighted text or replacing fraction strings with abbreviated characters (e.g., replacing "½" with "1/2") may also be implemented through similar AppleScript® programming routines.

As noted previously, the cleaning undertaken by the block 242 may vary depending upon which field or text box has been selected.

Before storing the cleaned highlighted text in a corresponding conversion field, the cleaned text is analyzed by a block 244 to determine whether it is valid. If the cleaned text is valid, then it is placed into the appropriate field or text box. If the cleaned text does not satisfy the text criteria an error message is displayed by a block 248. Control from the blocks 246 and 248 passes to the block 150 of FIG. 9 or the block 190 of FIG. 11.

While the process described above is an automated characterization and cleaning process, it should be noted that the user could manually enter data into the various fields or text boxes or could manually clean data stored in the conversion fields, as desired.

A detailed flow diagram of the process category selection block 158 of FIG. 9 is shown in FIG. 15. To input category information for a given object into the fields of the main dialog box of FIG. 2, the user clicks on one of the fields of the "Category Information" area, whereupon a block 280 of FIG. 15 displays a pop-up category dialog box (seen in FIG. 16). The category dialog box includes fields for indicating two different category assignments. A first set of fields (labeled "Main Category 1," "Sub Category 1" and "Detail Category 1") contains pull-down menus for entry of a first categorization of an object. A second set of fields (labeled "Main Category 2," "Sub Category 2" and "Detail Category 2") includes pull-down menus that allow the user to specify a second categorization for the object. Selecting one of the main category pull-down menus causes a block 282 of FIG. 15 to display a list of all possible main categories as specified in the categories file. Once a main category has been chosen, a block 284 causes the associated sub-category pull-down menu to display only those sub-categories that are stored in memory under the selected main category. The user may then select a sub-category by clicking on same. Similarly, once a sub-category is selected, a block 286 causes the detail category field pull-down menu associated with the sub-category to display only those detail categories stored in memory under such sub-category. At this point the user may select the appropriate detail category by clicking on such detail category. An OK button and a Cancel button are provided for exiting the category dialog box. Upon exiting the category dialog box, the selected category information is populated into the "Category Information" fields of the main dialog box as shown in FIG. 2.

FIG. 17 illustrates the programming executed by the block 162 of FIG. 9. This process identifies which high-resolution images from the page layout file are to be converted to a web-ready format and displayed on-line. The process could also convert these identified images into small thumbnail images and/or cropped images and could further store "beauty shots," which are the enlarged images shown after a user has selected a thumbnail image. At a block 310, if the user has clicked on one of the high-resolution images in the displayed page layout, and subsequently clicks on the "Image Name" field, a dialog box is displayed identifying the file name and giving the user the opportunity to change the file name. Once the user has indicated acceptance of the original or modified file name, a block 312 stores the name of the image file as a string in the "Image Name" field of the main dialog box of FIG. 2. Typically, the stored image name string will be manually edited to remove any name extensions from the name. If, for example, the cucumber image of FIG. 1 were stored as a Photoshop® image with the name Cucumbers.tif, the .tif extension which indicates that the file is a tagged-image file format file is removed. This is done in part to allow other name extensions, such as .gif and .jpg, to be appended onto the name upon image processing.

In addition, at any point during which the main dialog box of FIG. 2 is displayed and active, one or more image names identifying the image(s) derived from the high-resolution image can be stored in the fields under the "Image Names" heading of the "Detail Image(s)" area of the main dialog box of FIG. 2. Further, a description of each derived image can be inputted manually in the adjacent "Description" column of the main dialog box. A pull-down menu activated by clicking on the arrows of the button adjacent the label "Detail Image(s)" in the main dialog box can be used to select a "Counter" option to indicate the number of derived images stored for a given object. A button labeled "Edit Images" can be clicked on to allow editing of the description of the derived images in the "Description" field of the "Detail Image(s)" area of the main dialog box.

After all the appropriate fields of the main dialog box have been populated (e.g., as seen in FIG. 10), the operator can click on the "Save" button to save the data in the database file. The saving process is shown in FIG. 18. Every conversion field is stored in a field of at least one database record, where all database fields for a given object are relationally linked with one another. This process begins at a block 336 which checks to determine whether all fields which must be filled with valid data are in fact so loaded. If this is not the case, an error message is generated and the saving process ends. On the other hand, if this is the case, then a block 340 checks to determine whether there are any remaining fields or text boxes of the main dialog box to process. If so, a block 342 fetches the contents of the next field or text box of the dialog box. A block 348 then places the field value into the database. Control then returns to the block 340.

Eventually, all of the field values are loaded into the database, whereupon control passes from the block 340 to a block 352, which checks to determine whether there are any images to process. If this is true, a block 354 places the name of the image into the database and a block 356 writes out the name translation information pertaining to the image into the name translation file. Control then returns to the block 352. Once all the images have been processed, a block 358 clears out all fields (except as noted below) in the main dialog box and resets all counters. Control then terminates. The pull-down values for the main category, sub category, and detail category of the category dialog box are not cleared because they may be used while converting page layout information for the next object.

As should be evident from the foregoing, the programming of FIG. 18 stores cleaned data and image names into the database file. A sample database file product table having fields stored therein is shown in FIGS. 19A, 19B, and 19C. The data stored therein corresponds to the data in the conversion fields of FIG. 10. Thus, the number B-54031 is stored in the first record of the field "item number." Other values are stored at least some of the remaining fields of the database. In this regard, it should be noted that the database contains data that will not be displayed on-line or which is not necessary to identify images to be displayed. For example, a field "tax exempt" is loaded with data, as are the categories fields "main," "sub" and "detail," even though such fields are not required to permit on-line display. These fields are provided to allow a user to track such information for any desired purpose. In this regard, once the database is assembled it may be sent to the catalog publisher so that the publisher can track the information.

FIGS. 20A-20C illustrate the programming executed by the block 38 of FIG. 3. After all of the text information of the page layout file has been converted and cleaned, automated image processing functions are performed to convert specified high resolution images from the page layout file into images suitable for display in web browsers. The automated functions are undertaken by a program stored in a computer-readable memory, which may be the memory 26 of the computer system of FIG. 23. Alternatively, the instructions need not be stored in the same memory or run by the same processor that runs the conversion process discussed above. In fact, it may be desirable to have image processing performed on a different computer system than that illustrated in FIG. 23, although the computer system that undertakes the image processing would be similar or identical to that shown in FIG. 23.

Referring specifically to FIG. 20A, the same or a different user opens the image processing program, causing a block 380 to display an image dialog box illustrated in FIG. 21 and prompting the user to provide a specific location for the appropriate name translation file, (i.e., folder, directory or network location). Once this location is specified, a counter N is set equal to zero. (In the preferred embodiment the counter N is not used; however, employing the counter N in the description herein facilitates understanding. Also, some of the steps described in this and other portions of the specification do not find a one-to-one correspondence with the programming. (See the appendix appended hereto to obtain a full description of the programming.) Thereafter, a block 382 opens the name translation file which, as noted above, provides a mapping from the names of the high-resolution image files that appear in the page layout file to the names of the derived images that will be displayed on-line. Alternatively, as shown in FIG. 20, the image processing program could execute a set of instructions for automatically creating a first list of high-resolution image files stored in a particular location, and which further develops a list of thumbnail and cropped image names from the first list.

In either case, a block 384 opens a "cropped images file" containing a listing of the names of previously cropped images. If a cropped images file is not found (because it has not been yet created), an empty cropped images file is created at this point. This cropped images file may be stored at the location of the high-resolution files, or may be stored at another directory, folder or network location. A block 386 then compares the name N of the name translation file to the names stored in the cropped images file. If the name N of the name translation file is not in the cropped images file, then it has been determined that the image identified by the name N of the name translation file must be cropped and/or otherwise processed. Accordingly, a block 390 opens the high resolution image in a native image processing application, such as Adobe Photoshop®. As seen in FIG. 21, the image dialog box has five buttons: "Next," "Skip," "Start Batch," "Thumbnail" and "Quit." Because high-resolution images often cannot be loaded by a browser in a sufficiently short time and because of display resolution limitations, it may be desirable to crop the high resolution images and/or reduce the size in pixels of the images so that only a portion thereof will be displayed. At a block 392 (FIG. 20B) desired non-automated modifications to the high resolution image are undertaken by the user using the native image processing software.

After the desired non-automated modifications are complete, control pauses at a block 394 to allow the operator to select any one of the "Next," "Skip," "Start Batch," "Thumbnail" or "Quit" buttons. If the "Next" button is clicked on, then a block 398 executes specified changes to the image that can be programmed to be undertaken by the image processing application (i.e., the block executes "scriptable" or otherwise programmable changes to the image.) In the preferred embodiment, the block 398 converts the image file from a CMYK format (consisting of values for the printing primary colors of cyan, magenta, yellow and key (or black)) to a three-color RGB (e.g, red, green and blue) format suitable for computer monitor display. If desired, an indexed color format file could also be derived from the RGB format file. After programmed changes have been effected, a block 400 saves the modified image(s) as intermediate image(s) and names such files with a mapped name, which is mapped from the name of the high resolution image file. In the illustrated embodiment, the intermediates file(s), however, have no extension. The high resolution image file is then closed and a block 402 appends the name N to the cropped images file, indicating that the processing of this image has been completed. Control then returns to a block 430 of FIG. 20A.

If the block 396 determines that the "Next" button has not been selected, a block 404 checks to determine whether the "Thumbnail" button has been clicked on. If this is the case, a block 406 executes scriptable changes to the image (like the block 398) and a block 408 converts the image to thumbnail size. The high-resolution image is changed to thumbnail size through the use of image processing algorithms in the image processing software, which can (for example) reduce an 800x600 pixel image to a 40x30 pixel image. A block 410 then automatically performs an unsharp masking routine using the native application and a block 412 saves the resulting thumbnail image file(s) with mapped filename(s) mapped from the name of the high resolution image, but with the extension ".gif." Thereafter, a block 414 closes the thumbnail image file, a block 416 reopens the original image file and a block 418 appends the name N in the name translation file to an additional file called the "Done Images File," which lists all images that have been converted into a thumbnail. Control then returns to the block 392 of FIG. 20B to permit the corresponding high-resolution image to be processed in Photoshop® for custom modifications. This allows a set of custom modifications, separate from the custom modifications made in creating the thumbnail image, to be performed in creating the cropped image.

If desired, if an image name appears with a detail image extension, (i.e., if a file has a name with one of the extensions ".dn," where n=1, 2, 3...) then the programming could provide a dialog box indicating to the operator that the file is a detail image and, therefore, that no thumbnail image will be created by the programming of Fig. 22.

The above customizations of both the thumbnail image and the cropped image can be suspended by selecting the Quit button from the image dialog box. Specifically, if the block 404 of FIG. 20B determines that the "Thumbnail" button has not been selected, a block 420, FIG. 20C, checks to determine whether the "Skip" button has been clicked on. If this is the case, control returns to the block 430 of FIG. 20A. The block 430 increments the value of N and control passes to the blocks 386 and 388. At some point in the process all of the image files will have been processed and a block 428 passes control to a block 432, at which the automated processing routine of FIG. 22 is invoked. Following the block 432, a block 434 determines whether the user has selected an option "Open Folder" in the "Files" heading of a menu bar that is displayed by the operating system (Apple OS). If this has been selected, control passes to the block 380. Otherwise, a block 436 checks to determine whether the user has selected an option "Quit" in the "Files" heading of the menu bar. If this is true, program execution terminates. If this is not true, control returns to the block 434.

Referring again to FIG. 20C, if the block 420 determines that the "Skip" button has not been selected, a block 422 checks to determine whether the "Start Batch" button has been selected. If this button has been clicked on, then the user has requested that automated image processing be undertaken and hence the automated processing routine shown in FIG. 22 is invoked by a block 424. After the automated processing routine is complete, control passes to the block 434 of FIG. 20A.

If the block 422 determines that the "Start Batch" button has not been selected, a block 426 determines whether the "Quit" button has been clicked on. If this is the case, further program execution is terminated. Otherwise, control returns to the block 394 of FIG. 20B.

Referring now to FIG. 22 the automated procedures can be facilitated by an automation tool, such as PreFab Player® from PreFab Software, Inc. of Westford, MA, which allows the execution of pre-defined actions such as button selections, keystrokes, and control commands. Preferably, the automated procedures at least open each RGB file from the cropped files list, apply an unsharp masking to the image, apply an image compression such as a JPEG compression to the image, and save the compressed file in the compressed image format. If the file is not already in the "Done Files" list, the same RGB file is then re-opened, scaled to thumbnail size, sharpened, converted into an indexed color file and saved as a compressed image file, such as a GIF file, after which the file name is stored in the "Done Files" list. Other automated actions may be carried out in a similar manner. The thumbnail and cropped images created in this manner and stored in the proper location correspond to the image names stored by the programming of FIG. 18. Furthermore, the images are sized and image processed for proper display in the on-line catalog.

Specifically as seen in FIG. 22, a block 450 reads the next name from the cropped images file (if this is the first pass through the programming of FIG. 22, the block 450 reads the first name in the file.) A block 452 then checks to determine whether all images in the cropped images file have been processed. If so, then execution of the programming of FIG. 22 ends. If not, a block 454 performs unsharp masking on the file contents and a block 456 performs a JPEG compression and saves the resulting file with the extension ".JPG." A block 458 then determines whether the file is a detail image. If this is not the case, a block 462 checks to determine whether the file name is in the Done Files list. If this is also not the case, a series of blocks 462, 464, 466 and 468 changes the size of the image to thumbnail, performs unsharp masking, executes scriptable changes (identical or similar to the block 398 of FIG. 20b) and saves the resulting file(s) with the extension ".GIF." A block 470 then closes the image file(s).

Following the block 470, or following the blocks 458 and 460 if the file is a detail file or if the file is in the Done Files list, a block 472 deletes the intermediate file(s) (i.e., the file(s) that were stored with no extension). A block 474 then determines whether the Done Files list includes the file name. If this is not the case, a block 476 adds the file name to the Done Files list and control returns to the block 450. The block 476 is skipped and control returns directly to the block 450 if the block 474 determines that the Done Files list does not include the file name.

It should be evident from the foregoing that the process and system described hereinabove coverts the unstructured and untagged data in the page layout file into searchable data related to an object.

With reference now to FIG. 25, the above-described system may be incorporated into a document production workflow 500 that streamlines the process of modifying one or more page description files 502 of a first book for the printing and distribution of a second, customized book (not shown). The page description files 502 should not be limited to any one particular page description format, such as QuarkXPress® as specified by Quark, Inc., but rather shall be understood to include any type of page mark-up or layout file (*e.g.*, Acrobat® PDF as specified by Adobe, Inc. or HTML), or any other type of file that comprises content and layout information for one or more pages to be printed, displayed, or otherwise distributed. It should further be understood that each page description file 502 may be representative of any number of pages to be printed, such that an entire book may be represented by a single file. It should be still further understood that the term "file," as used herein, includes any computer-readable collection or assembly of data and/or information in either a localized or distributed manner, regardless of the manner in which the data or information is accessed.

Each page description file 502 may, for example, constitute a QuarkXPress® file representative of one or more pages of a catalog to be distributed to consumers to provide product-related information on a seasonal basis. Thus, the page description file 502 may, for instance, comprise product information for certain fashion items directed toward the current season. As will be explained further hereinbelow, the content and page layout data stored in the page description file 502 leads to the production of a first catalog 504.

Practice of the present invention, however, is not limited to the specific example of catalog production, and is broadly applicable to any type of book or document production workflow in which customized printed matter is produced from previously designed matter. Nevertheless, it shall be hereinafter assumed for purposes of illustration only that the printed matter to be produced constitutes such clothing catalogs.

In accordance with one embodiment of the present invention, the document production workflow 500 modifies the content and page layout data stored in the page description file 502 to produce a further page description file 506 for production of the second, customized catalog. In one instance, the second catalog may be customized to reflect inventory changes since the distribution of the catalog 504.

Initially, the page description file 502 representative of the first catalog 504 is generated in a page make-up software program or system (such as the system described hereinabove in connection with FIG. 23) after certain pre-press steps 508 have been performed. The details of both the pre-press steps 508 and the page make-up process are well known to those skilled in the art and any one of a variety of known systems may be employed. In one embodiment, the page description file 502 is developed as a QuarkXPress® file that can be raster image processed to develop bitmap representations of each page of the catalog 504. These bitmap representations may be subsequently used for rendering of the pages on press cylinders or plates 510 in connection with printing using conventional presses or, alternatively, for rendering of the pages using a digital press. In any event, a press generates multiple copies of the catalog 504 having one or more pages. In a digital press embodiment, the digital press responsive to such bitmap representations may be utilized to print copies having further customization, as explained hereinbelow.

Next, the printed copies of the catalog 504 are then provided to recipients in a conventional distribution process 512, the details of which are not pertinent to practice of the present invention. The recipients of the catalog 504 then place orders for items described in the catalog 504 via mail, telephone, and the like. These orders are handled manually or, preferably, by a processing system 514 having one or more computer systems that may be localized or distributed in nature and, more generally, may take on any suitable form.

At the same time that the above-described conventional catalog order process is occurring, web-based commerce based on the same page description file 502 is also resulting in orders for the processing system 514. More particularly, the page description file 502 is processed in accordance with the system and method described hereinabove to generate one or more web pages 516 for display and delivery via the Internet or other network as an electronic document. Briefly, the page description file 502 is first processed in a block 518 to generate a database for storing data indicative of portions of the page description file 502, which may constitute textual, image, or line art items. For example, if one of the pages represented by the page description file 502 contains a textual portion comprising a paragraph of text describing a certain product, the database generated by the block 518 may contain a record for the certain product with a field designated for storing product descriptions. The database may correspond with the product table shown in FIGS. 19A-19C.

More generally, the block 518 is preferably implemented by the software routine set forth hereinabove that allows a user to characterize each portion of the page description file 502. In summary, the block 518 implements the steps described in connection with FIG. 24 and other related figures such that the content (*e.g.*, image or text) data are then extracted from the page description file 502 to generate the database. For purposes of clarity in explanation only, the database generated by the block 518 will be hereinafter referred to as the "catalog database." As set forth hereinabove in connection with the product table of FIGS. 19A-19C, the catalog database has a plurality of records wherein each record is associated with one or more products or items of the catalog 504. In one embodiment, each record of the catalog database stores detailed information for a number of products corresponding with the number of products to be displayed on each page of the catalog.

As described hereinabove, the extraction of content data from the page description file 502 also results in the generation of derived image files that may be, for instance, cropped, of a low resolution quality, and/or of the "thumbnail" variety. These images are particularly well-suited for use in connection with the web pages 516, and may be referenced by filename in the catalog database. However, these image files are generated at a block 522 that may also generate high resolution images suitable for printed publications. The manner in which the processing of the page description file 502 generates high resolution images saved as separate files (rather than the low resolution images, for instance) will be readily apparent to those skilled in the art in light of the description hereinabove of the web-based conversion system. The image data associated with these high resolution images may then be incorporated into the page description files 506 developed by the document workflow system 500, as set forth hereinbelow.

The catalog database and any associated image files are then provided to a network server 520 to generate the web pages 516 for transmission via the Internet, an Intranet, or any other type of distributed or localized network. The web pages 516 may provide an electronic version of the content provided via the printed catalog 504, or some other version modified to suitably display via the Internet. In any event, the transmission of the web pages 516 results in further orders for handling by the order processing system 514.

As a result of the orders handled by the order processing system 514, the catalog database may need to be modified. For example, if a certain product is no longer available in a particular color or size, the content presented in the second, customized catalog should be modified to reflect this unavailability. To this end, the catalog database is provided to a block 524, together with data representative of, indicative of, or relating to these database modifications, for analysis and processing. The data utilized by the data analysis block 524 need not be limited to data relating to orders that have been processed, but may further include questionnaire data provided from a block 526 or any other data that may be utilized to analyze whether customization of the content of the catalog should occur.

In a preferred embodiment, the data analysis block 524 not only modifies the catalog database in light of orders processed to date, but also performs data mining routines known to those skilled in the art, such as product or affinity clustering, regression analysis, correlation, positive and reverse seasonality, and the like. In this manner, the catalog database may be modified to reflect actual sales (*i.e.*, in terms of which products are still in stock), as well as data related to actual sales, such as whether sales of two or more items have exhibited a correlation. Analyzing such related data may then lead to a customized catalog with a different layout designed to exploit beneficial relationships between items.

The data analysis block 524 may be implemented via a software routine that provides a user interface for displaying data collected from various sources, such as questionnaires, or data that has been generated from actual sales data. The data may then be analyzed by an operator or user as appropriate by, for example, compiling and comparing sales statistics of two or more products to check for correlation and the like. Alternatively, the data analysis is either performed or assisted by a data mining application or suite of tools known to those skilled in the art. Subsequently, the user interface may provide an option for reordering the records in the catalog database in light of the correlation or other statistics. For example, if the data collected by the block 524 shows that two clothing items have been sold in complementary fashion, the catalog database may be modified to reorder the records in a manner such that the page description file 506 for the second, customized catalog to be printed has the complementary products presented on the same page or in similar sections of the customized catalog. As a further example, when an item has sold out of a particular color or size, the corresponding information associated with that color or size is removed from the catalog database. Thus, when the catalog database is utilized to generate the page description file 506 for the second, customized catalog to be printed, the content of the catalog will reflect the most recent evaluation of the inventory situation.

The manner in which the catalog database determines the content data for the page description file 506, and thereby the second, customized catalog to be printed, will now be described. In general, the page description file 506 is generated from the catalog database (including any modifications thereto by the block 524) and a template or series of templates (hereinafter "catalog template"). In one embodiment, the catalog template is derived from the page description file 502 for the previously printed catalog 504. Alternatively, the catalog template is developed independently of the page description file 502. In either case, the catalog template comprises placeholders that establish or define an area or position of the page to be populated by the content data stored in the catalog database.

Preferably, the catalog template is generated in a page make-up software application, such as QuarkXPress® or the like, that has been modified (via a software extension or otherwise) to support placeholders.

The modified QuarkXPress® program may be adapted for operation on the Apple Macintosh® operating system or any other operating system, such as the Microsoft Windows® operating system. The details of the programming for use as a QuarkXPress® extension to create a template or template file having placeholders may be found in commonly assigned, co-pending application Serial No. 08/498,397, the disclosure of which is hereby incorporated by reference herein.

A placeholder, as used herein, should be understood to establish an absolute or relative position on a page where variable information is to be printed. That is, each placeholder in a template will be populated with variable information such that the variable information is incorporated into the resulting page description file in preparation for printing or distribution, in general.

In general, the modified QuarkXPress® program provides a user interface for establishing placeholders in the template. The user interface, in turn, is generated via the programming described in the above-referenced application and will not be further discussed herein.

In one embodiment of the present invention, the page description file 502 is opened into the modified QuarkXPress® program in a block 528 for conversion of the portions thereof into placeholders. As described in the above-referenced application, one or more of the image portions set forth in the page description file 502 may be replaced by reference images associated with a particular field in the catalog database that indicates image file names for the images generated by the block 522. Likewise, any textual portion may be replaced with a placeholder that calls out to a database field name for which text has been saved in the catalog database during the block 518. In this manner, one or more pages of the catalog 504 based on the page description file 502 can be converted into one or more templates for automated incorporation of variable information.

After converting the appropriate portions of the page description file 502 into placeholders, a catalog template is generated by a block 530. The block 530 preferably saves data indicative of the catalog template and, thus, the placeholders defined therein, in a template file.

In an alternative embodiment of the present invention, the template file has either already been generated in connection with the page description file 502 or has been generated independently thereof. In either of these cases, the template file is merely retrieved by the block 530 for processing.

Generation of the catalog template in connection with the page description file 502 may be useful when a great deal of information or content will remain constant between the catalog 504 and the customized catalog to be printed. For example, the template may retain a background image portion from the page description file 502 when each page of the catalog 504 contains a fixed image that forms a background for all of the other portions on the page. Conversely, generation of the template in an independent manner may be useful when the customized catalog to be printed has a completely different layout than the catalog 504, such as a different number of products displayed on each page, a different background, a different header or footer, and so on. These differences may be particularly acute when the customized catalog to be printed constitutes a clearance catalog.

In one application of the present invention, a clearance catalog to be printed comprises a cover page followed by a plurality of pages of the same basic layout. In this case, two templates may be generated by the block 530 independently of any prior page description file. More particularly, a template having mostly fixed information is generated for the cover page of the clearance catalog. The fixed information may include content information for identifying the source of the catalog, a cover image, text specifying applicable dates for transactions, and text identifying the catalog as a "Special Clearance Catalog."

In this particular example, each page of the remainder of the customized catalog, *i.e.*, the body of the catalog, follows the same basic format and layout. As a result, each one of the body pages may be derived from the second of the two templates, hereinafter referred to as the "body template." The body template may include one or more placeholders for product images, together with one or more corresponding textual sections for the respective products associated with the images. The textual sections, in turn, may comprise a placeholder for a general product description, followed by placeholders for the available colors and sizes. The textual section may further include one or more additional placeholders for such content as SKU number, original price, clearance price, and the like.

In this case, the body template is associated with a plurality of pages of the customized catalog to be printed and, thus, may be utilized multiple times during generation of the page description file for the catalog. It should be noted, however, that the present invention may be practiced using any number of templates with each respective template being associated with any number of pages in the customized catalog to be printed. To that end, each record in the catalog database may include data indicative of which template of a plurality of templates should be used for the content data stored in that record. This template-identifying data may be entered by an operator of the above-described system for extracting the content data from the page description file 502. Alternatively, the block 524 may include conditional programming that analyzes the content data for a particular record to determine which template should be associated with the record. For example, the catalog database may include statistics regarding whether small or large images have resulted in purchases of the particular product. Based on the statistics, a particular template may be selected based on the size of an image placeholder therein. Similar schemes for performing such conditional layout design may be devised by those skilled in the art based on the data analysis performed by the block 524.

Once the template or templates have been either generated or retrieved by the block 530, the page description file(s) 506 for the customized catalog to be printed are generated by a block 532. The block 532 is generally responsive to the catalog database (as modified by the block 524), any images files generated by the block 522, and the catalog template(s) generated or retrieved by the block 530. More particularly, the block 532 processes the catalog template(s) in accordance with the content data stored in the catalog database that has been associated with the template(s). That is, for each record in the database, a selected template is populated with content data identified by the database record in accordance with the placeholders in the template. Further details on the programming routines necessary for processing the template(s) are set forth in the above-referenced application and will not be repeated herein. However, the programming routines are preferably modified to provide a single output file for each populated template. In other words, the information provided by the template and the database is not processed to develop separate streams of page description data (*e.g.*, a master file representative of fixed information, and a variable data file representative of information that varies from record to record). Rather, the block 532 preferably writes out a single page description file having the fixed information from the template as well as the content information provided by the database record being processed. To this end, the fixed information and the placeholder information in each template need not be separated (as it is done in the above-referenced application). Instead, a copy of the catalog template may be generated and processed to populate the placeholders with content data. However, if the customized catalog will be printed using a digital press, the block 532 may, in fact, separate the output into two or more data streams, such as a master data stream and a variable data stream.

It should be noted that the catalog database generated in accordance with the teachings of the present invention may have to be modified by the block 524 in form (rather than content) to be compatible with the database format specified by the processing system of the block 532. It should also be noted that the filenames generated by the block 522 for the image files should be compatible with the filename convention specified for the processing system of the block 532.

The block 532 generally provides an initial dialog box for initiating the process of generating the page description file 506. In addition to providing a standard "START" or "RUN" option, the dialog box allows a user to define parameters for the job to be run. More particularly, the user has the option of selecting one of several different types of output formats (*e.g.*, QuarkXPress®), as well as whether the output should be dumped into one single page description file or multiple files, which may, for example, result in one file per page (or flat) of the catalog. The dialog may also prompt the user to identify the catalog database by name and location (*i.e.*, path), as well as the total number of pages in the catalog.

After the page description file(s) have been generated, an imposition routine may be run to set forth the pages in proper order for printing. In one embodiment, imposition may be handled by an off-the-shelf software package for imposing one or more page description files. Alternatively, if the catalog will be printed using a digital press, the imposition routine set forth in the above-referenced application may be utilized to impose the pages of the customized catalog.

As a result of the processing of the block 532, one or more page description files 506 may be provided to a conventional computer-to-plate press system or a digital press for production of a customized catalog. If a digital press is utilized, the catalog may be further customized by treating the page description file(s) 532 as template files representative of fixed content for a plurality of pages. The system of the above-referenced application may again be utilized to add further placeholders to the layout defined by the page description files 532. These placeholders may be associated, for instance, with recipient address information for a cover page. Then a variable information database (such as that described in the above-referenced application) having addressee information and other variable information could be used to print catalogs with content customized for each recipient of the catalog, as specified by the template(s).

It should be noted that the catalog database may include additional data directed toward this added layer of customization. In such an embodiment, the data analysis block 524 may include functionality for providing a user with an opportunity to modify the catalog database appropriately.

The document production workflow described hereinabove shall be understood to include a software system or suite of software applications that may or may not be incorporated into a single, cohesive application. In any event, the workflow is practiced using a software system having routines dedicated to implementing the routines specified hereinabove. The routines, in turn, are stored in a computer-readable medium, which may comprise any type of computer-readable storage mechanism, such as an optical or magnetic disk. Alternatively, the computer-readable medium may form a part of a publishing or production system such that the routines are stored in resident memory of the system, or in a portion of the system that stores data optically, magnetically, or in any other fashion known to those skilled in the art. Still further, the routines may stored in either a localized or distributed manner such that the computer-readable medium may comprise a network of computer-readable media.

Numerous modifications to the present invention will be apparent to those skilled in the art in view of the foregoing description. Accordingly, this description is to be construed as illustrative only and is presented for the purpose of enabling those skilled in the art to make and use the invention and to teach the best mode of carrying out same. The exclusive rights of all modifications which come within the scope of the appended claims are reserved.